A Group-Oriented Framework for Coalitions

Eric Hsu

Artificial Intelligence Center
SRI International
333 Ravenswood Ave. Room EJ211
Menlo Park, CA 94025
hsu@ai.sri.com

Abstract. Coalitions exemplify the fundamental challenge of inducing coherent group behavior from individualistic agent structures. The Collective-Agents (CA) framework rejects the distinction between individual and group deliberation, on a functional basis. Acknowledging that a group does not think using a brain, and an individual brain is not divisible into multiple minds, the CA framework nevertheless seeks an analogous correspondence between the intentional attitudes of individuals and groups alike. Resulting agents are extremely elegant, allowing hierarchical decomposition of coalitions into sub-groups and providing savings in communication costs. More importantly, such principles allow the use of abstract software wrappers for transferring advances in individual planning, control, and scheduling directly to a group setting. After formalizing the framework, we will demonstrate such claims in principle and in an implemented system.

1 Introduction

The fundamental challenge of constructing coalitions can be expressed as the transition from individual to group action. Agent architectures based on beliefs, desires, and intentions ("BDI architectures") are particularly well-developed for individual agents, but such mental constructs have defied easy translation to groups. In practice and in principle, most computer scientists and philosophers are skeptical of collective intentional attitudes, on the grounds that minds must be individual and indivisible (Bratman 1992;Grosz & Kraus 1996.) Hence, research has focused on applying individual attitudes to group content: agents must develop communicative protocols for sharing their beliefs, promoting their own desires, and cultivating intentions to perform roles in teams.

The Collective-Agents (CA) framework rejects the distinction between individual and group deliberation, on a functional basis. Certainly a group does not think using a brain, and an individual brain is not a congress of disparate voices. However, individuals and groups might employ analogous processes that operate equivalently over corresponding intentional attitudes. The CA framework seeks to establish such correspondence by using the same logical constructions to express the desires, intentions, and actions of individuals and groups alike.

As a result, CA implementations are extremely elegant, employing the same high-level data structures and algorithms for single agents as for networks of agents. This allows for hierarchical decomposition of teams into sub-teams, and recursion through multiple layers of planning and action. At the same time, communication costs can be greatly reduced. While the benefits of such a straightforward approach might be clear to the distributed or parallel computing communities, it might appear quite unprincipled to anyone who is familiar with traditional BDI architectures where individuals and groups are quite distinct.

To that end, any introduction to the Collective-Agents framework should begin by explaining the intuition behind viewing groups as individual entities and individuals as group-like constructs. The next step will be to present the framework itself, followed by a more concrete justification based on its adherence to such intuitions. After potential intuitive objections are addressed, CA can be further characterized by comparison with existing approaches. Before concluding, we will present our generalized CA implementation and evaluate it in a sample domain.

2 Background

Groups When the Dean of a university asks how much money the Computer Science department intends to budget for hardware purchases, no particular professor has the discretion to decide for everybody.[1] Traditional individualistic formalisms (Grosz and Kraus 1996) might state that a professor "intends that" the budget be set at some level. Yet if the final figure is a compromise then we cannot say that any one of the professors intended that the department thus allocate its resources. It would seem that their original preferences were more akin to desires

[1] This scenario is a combination of examples due to David Velleman and to John Searle.

than intentions; if the department's spending depends on any entity's intention, why not the department's?

To speak of a group, in and of itself, holding an intention raises the question of how such intentions could be formed, and then executed. While formal groups like faculties might obey explicit constitutional rules for transforming individual desires into courses of action, more casual situations like organizing a departmental party might be decided by very arbitrary means. Likewise, financial agreements might be so explicit as to precisely specify individual courses of action, while a mandate to hold a party might leave individuals mostly to their own devices. To make an unambitious generalization, intentional group action can be seen as a function mapping various individual desires into a cohesive aim, together with another function that apportions the various individual roles that will achieve that goal.

Individuals The Collective-Agents formalism aims for efficiency by treating individual agency as an analogous process, allowing full integration between agents and their groups. Hence, an individual agent operates as a composite of the roles it plays in various groups, characterized by processes that mediate between such roles and integrate them into a decisive course of action. For instance, being a Computer Science professor means playing the roles of instructor, administrator, and researcher, among others (Sonenberg 1994.) This means coordinating conflicting goals generated by these roles; any given afternoon might require holding office hours, attending a departmental meeting, and writing a conference paper.

More analytically, the distinction is between what Sellars calls the "plain I" and the various "I_{we}"'s which pursue various group activities[2] (Sellars 1968.) A professor who skips a conference in order to prepare a lecture might explain, "The teacher in me got the better of me." This is not to argue that human brains are divided into multiple sub-brains. Rather, CA relies on a conceptual scheme that represents the coordination of an agent's commitments by correspondence with the roles that generate such goals.

3 Toward A New Framework

Individualistic Alternatives At a crudely abstract level, most BDI architectures for individual agents resemble the following loosely-defined syntax:

Agent ::= (*Arbitrator*, *Mental-State*)

Arbitrator ::= (*Planner*, *Executor*)
Mental-State ::= {beliefs, desires, intentions, etc.}

Planner $\subset$ {(*Mental-State*, *Plan*)} (a functional relation)
Executor $\subset$ {(*Plan*, actions affecting the world)} (a functional relation)

Hence an agent's actions are a function of its plans, which are themselves a function of its current mental state. Several important details are left out of this sketch. Beliefs must be incurred by perceptive processes, and interact with the other mental information. The planner and executor should operate concurrently; changes in the agent's mental state can affect its plan, and hence its course of action. However an architecture addresses such complexities, though, it does so using these operational modules, or equivalent models.

Many multi-agent systems are reluctant to depart from this framework, and use the same structures to implement group activity as depicted in Figure 1. In the figure, "Group 1" does not exist as a computational entity, so much as it results implicitly from the communication between Agents A and B. In order to deliberate concerning their group as a whole (for instance, to decide whether to serve as a sub-unit in a larger group,) the agents must explicitly refer to "Group 1" in the contents of their communications.

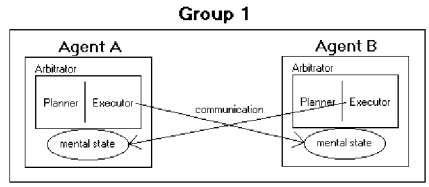

Figure 1: A Two-Member Group of Individualistic Agents.

[2] For instance, a professor might harbor an "I_{we}" which participates in such declarations as "We the faculty intend to alter the budget."

Figure 2 illustrates two forms of complexity that arise from such amorphousness. First, the addition of a third agent multiplies the number of communicative channels. In general, the number of possible pairings increases quadratically with each additional agent, endangering communication bandwidth. Secondly, Groups 2a and 2b must be specifically disambiguated within messages since the individualistic agent architecture does not automatically capture hierarchical group structures. That is, there is no architectural difference between 2a, a two-entity group whose first member is itself a two-entity group, and 2b, which is a three-entity group. At an implementational level, such generality should not be mistaken for flexibility. For instance, for both groups to function simultaneously, Agent A must index its correspondence with B and C by relevant group for lack of inherent context.

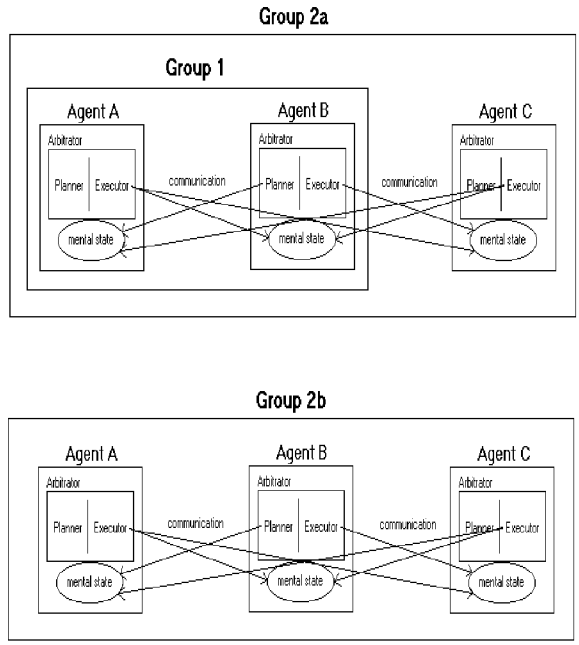

Figure 2: Individualistic Groups of Greater Complexity.

The Collective-Agents Framework The observations made in Section 2 suggest that group-orientation can relieve the above problems, and provide some additional benefits. The following syntax outlines the basic structure of Collective-Agents (the '+' symbol designates "one or more"):

Collective-Agent ::= *Individual* | *Group*
Individual ::= (*Arbitrator*, *Individual-Role*+)
Group ::= (*Arbitrator*, *Collective-Agent*+)

Arbitrator ::= (*Planner*, *Executor*)
Constituent ::= *Collective-Agent* | *Individual-Role*
Individual-Role ::= one of the roles which an individual plays

Planner ⊂ {(*{Report+}*, *Plan*)} (a functional relation)
Executor ⊂ {(*Plan*, *Instruction*+)} (a functional relation)

Report ::= (*Constituent*, {beliefs, desires, intentions, etc.})
Instruction ::= (*Constituent*, actions affecting the world)

Hence the same arbitrating structure governs groups and individuals alike, allowing hierarchical decomposition of teams, and centralizing deliberative processes for team-members. Figure 3 presents the same two group structures from Figure 2 as represented using the Collective-Agents framework. C-Agent 2a is a two-member entity, whose first member is a team composed of C-Agents A and B. Within this structure, neither A nor B communicates directly with 2a; the arbitrator for C-Agent 1 acts as an intermediary instead. C-Agent C, which is not involved in the activities of C-Agent 1, only communicates with A and B concerning the affairs of C-Agent 2a, thus speaking solely through the arbitrator of that group.

Unlike other "collective" agent architectures, this one does not distinguish between individual or collective arbitrators, meaning that they function identically.

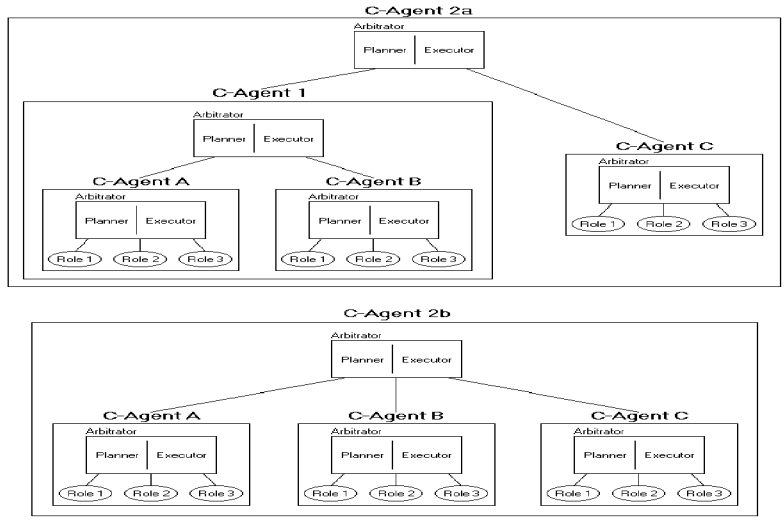

Figure 3: Two Complex Collective-Agents

4 Benefits

Communicative Efficiency By imposing such rigid structure upon agent interaction, the CA framework can improve efficiency for a large class of group topologies. Table One summarizes some easily observed characteristics resulting from a tree-structured, CA-based coalition structure.

	CA Best Case	CA Worst Case	Individu-alistic
Arbitrators	$O(n)$	$O(n)$	$O(n)$
Channels	$O(n)$	$O(n)$	$O(n^2)$
Hops	$O(1)$	$O(n)$	$O(1)$

Table 1: Comparison of Computational Complexity for Groups Assembled from n Individuals.

These observations measure three types of cost. The first is the number of arbitrator processes that would have to be run for a group of n agents. While any arrangement of n individualistic agents would merit n arbitrators, the number varies for Collective-Agents based upon their structure. In the best case, the n individuals would simply comprise a single group, and n+1 arbitrators would be necessary—one for each individual and one for the entire group. The worst case is based on the limitation that a group must contain at least two collective-agents; otherwise a single individual could spawn an arbitrary number of arbitrators if it were nested deeply enough as a subgroup of a subgroup of a subgroup, and so on. Accordingly, the worst-case structure is a balanced binary tree of Collective-Agents with the n individuals as leaves. Even so the number of arbitrators would be just 2n-1.

The second measure is the number of communicative channels that must be available between pairs of arbitrators. While this might not be significant in laboratory simulations, an agent deployed in practical applications cannot always communicate with every other agent without a cost, if at all. Just as the Internet's hierarchy of gateways, routers, and backbones counters the impossibility a running cable between every pair of computers in the world, CA hierarchies enable efficient communication between individuals. There are $(n^2-n) / 2$ potential pairs within n agents. For each individualistic agent within a group to be able to broadcast to all its partners, a channel must exist for each pair. On the other hand, a CA individual within the best-case structure already described would need only report to the group arbitrator, which would then pass the relevant information back down to the other individuals. This would require n channels, while the worst-case scenario (also a binary tree) would rely on just 2n-2 channels.

The disadvantages of such parsimony affect the third area. Just as IP packets must make a number of hops across the Internet before arriving, Collective-Agent communications might pass through a number of managing arbitrators before reaching interested parties. Given the previous assumption that CA groups must contain at least two members, the maximum number of hops occurs in the deepest possible binary tree with n individuals as leaves. In such a case information from agents in the most deeply nested group must traverse n channels before reaching the top-level group and its individual member. However, in best-case hierarchies, which consist once again of single groups, any message must make at most two hops. While ubiquitous channels between individualistic agents provide one-hop connections, the complexity introduced by CA is at worst linear, and constant at best.

Abstraction Beyond such metrics, the hierarchical decomposition employed by C-Agents can provide additional implementational advantages akin to those offered by functional decomposition in programming languages. Since Arbitrators can govern individual roles, or other C-Agents alike, they can be implemented using the same computational structures. "Belief-Desire-Intention" architectures are well developed for individual agents, and can be applied directly to C-Agent groups once communication between agents is encapsulated as mental events within the collective whole.

For instance, when two CA partners report conflicting desires during negotiation, their arbitrator can reconcile this difference in the same way that traditional individuals would make decisions based on conflicting goals. Where before the two desires would reach the planning module directly from an agent's mental repository, now they would arrive as communications from two constituent C-Agents. While an individual executor would directly instigate agent actions, now a CA arbitrator sends instructions to its subordinates.

This is not just an implementational shortcut, but a new way to formally conceptualize and then implement at worst the same patterns of agent coordination. If two traditional collaborators devote some percentage of their processing and messaging to negotiate a plan for their shared activities, only conceptual prejudice prevents us from extricating such activity into a new process. If two conventional "collective" agents seek to centralize their negotiations through an intermediary server, why should this server's operation deviate from existing well-developed individual planning methodologies?

Because C-Agents can be generated dynamically from heterogeneous classes of arbitration schemes, this framework is not a design document for agent interaction, but for a wrapper system to generate such dynamics from individual architectures. This point is best understood through the implementation and its use in a sample domain. sample domain.

5 Implementation

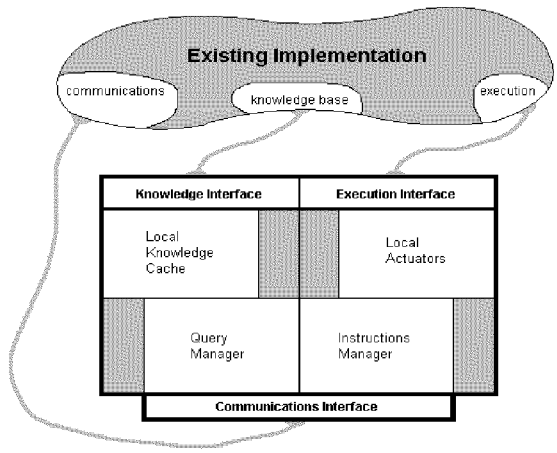

Figure 4: C-Agents Wrapper Design.

The C-Kit The generalized C-Agents implementation (named "C-Kit") is a domain-independent control system that operates in a self-contained thread. In addition, it is installed independently from whatever existing planning and execution system a researcher is using, connecting in only three places through a highly abstract interface. In short, it constitutes a software wrapper for turning any BDI agent into a C-Agents arbitrator that can interact with other, potentially different, agent implementations that have also been outfitted with the C-Kit.

Figure 4 illustrates the operation of such a composite agent. Here a generic agent process has been outfitted to serve as the arbitrator of a group of C-Agents. Mirroring the formal pairing of deliberator and executor, the C-Kit provides two principle services to the pre-existing implementation: the Knowledge Interface and the Execution Interface. Where an individual planning agent formerly consulted its own knowledge base through the course of deliberation, it now consults the C-Kit's knowledge base, which compiles individual beliefs from the constituents. A query first passes through the local knowledge cache to see if it can be answered by already compiled information. If not, it goes through the query manager and is communicated to subordinates. Likewise, the query manager is also responsible for responding to queries from superiors, potentially passing such requests on to its constituents. The specific methods for resolving conflicting reports or deciding what information to cache are determined by the domain and not specified by the C-Kit. In the test domain presented below we coded simple voting procedures and heuristics, but the C-Kit should be seen as a workbench for encoding more advanced methods.

Similarly, a C-Agent's actions are actually executed by actuators controlled by its constituents. Hence, when the

group wishes to perform a certain series of actions, the commands actually go through the Execution Interface, which first checks to see if any can be performed by local actuators. The Instructions Manager dispatches any remaining orders to constituents. In the other direction, it receives instructions from superiors and either executes them locally or dispatches them to constituents. Representing a group entity's capabilities as a compilation of constituent capabilities and their possible interactions is again left to the domain expert. Where before such models were installed into each necessarily identical individual agent or emerged implicitly from hard-wired behavior, they can now be deployed once within the C-Kit and used with heterogeneous collections of agent implementations.

The third and final interface, depicted at the bottom of the figure, governs the messaging activities produced by the other two. Specifically, the Communications Interface sends queries and instructions to constituents, and replies to queries from superiors. Similarly, it queues query responses from constituents, and queries or instructions received from superiors. To do so, it interfaces with whatever communicative facility the existing agent implementation already uses. If there are none, then certainly one must be created in order to do multi-agent planning, and then it should be connected directly to the C-Kit through this interface.

The toolkit is implemented using the Java language "Remote Methods Interface" package, which allows an agent to perform operations on a remote host without relying on high-level message passing. This way, if an agent is already deployed it can interact with a C-Kit running on a different host almost transparently. In Figure 4 it is the gray connectors that signify this link. Later, including RMI in the design should allow for mobile code, whereby arbitrators will be able to transfer themselves to new hosts in search of extra processor time or faster access to local actuators. However, such functionality has yet to be implemented.

Before continuing a final observation should be made concerning the domain-specific components of the knowledge and action interfaces. In particular, the compilation of parent-agent beliefs and capabilities is left open in this particular implementation, but related research efforts toward similarly aligned goals may provide generalized procedures for doing so. Specifically, the area of structured theorem proving seeks to perform inference over multiple knowledge bases, while one main area of Semantic Web research is automated service compilation from multiple sources. The author is most familiar with work at the Stanford Knowledge Systems Laboratory (Amir & McIlraith 2001, McIlraith et al 2001).

6 Example Scenario

Suppose that in some future military scenario, two autonomous ground vehicles G1 and G2 patrol for enemies within a sensitive area, while a pair of autonomous air vehicles A1 and A2 patrol the perimeter. If either party detects an enemy, the first priority is for the detecting team to pursue it immediately, the second is to maintain the inner patrol, and the least important is to continue the perimeter patrol. Also, suppose that a single air vehicle can patrol an area or pursue a target, but such tasks require two ground vehicles working together.

Based on these mission criteria, the desired outcome depends on whether the target appears before the ground or air vehicles. If it appears within the area, both ground vehicles should immediately pursue the target, with an air vehicle taking over the inner patrol. On the other hand, if it appears on the perimeter, only one air vehicle should pursue, as the other continues the perimeter patrol and the ground vehicles continue their inner patrol.

In an individualistic agent system, such conditional behavior would have to be distributed across each agent. The initial patrol plan must specify each agent's new role given each contingency, or alternatively the agents must re-negotiate their tasks upon detecting the target. Both approaches elicit broad interest within the agent systems community (with Ortiz, Hsu et al. 2001 and Ortiz & Hsu 2002 representing each approach within this same type of scenario.) On the other hand, the CA framework would encapsulate such group-level decisions within new computational structures.

Specifically, each agent can be wrapped using the C-Kit and connected to one of two independent C-Agents representing the patrol teams, P1 and P2. Further, the two teams would in turn connect via a third new C-Agent C1 representing the overall coalition. It is easy to see that if these vehicles and their mission were part of an even larger mission, then C1 could connect to an even higher agent. Each of the non-leaf C-Agent consists of an unmodified individual agent implementation outfitted with the C-Kit so that it operates as an individual but serves as a group.

Thus, it is C1 that receives a reported detection over the chain of communications, and dispatches a pursuit instruction to the nearer of P1 and P2, while assigning the inner patrol task to the other. It is important to stress that the individual agent operating C1 does not "realize" that it is running a group. It thinks it has two sets of actuators (P1 and P2) and is using them both to patrol. The reported detection arrives in its knowledge base the same as if C1 had sensed it directly, and its subsequent instructions are translated by the C-Kit from calls to its supposed actuators.

One consequence is that C1 does not specify that the air vehicles should split up when they are closer to the target. Rather, the C-Agent representing their partnership (P2, without loss of generality), receives the instruction to pursue the target, and dispatches it to either A1 or A2. Again, the individual agent implementation has been initialized with two sets of actuators, by virtue of the C-Kit. At first P2 uses both for its single task of patrolling, and patrol instructions are dispatched to the C-Agents representing A1 and A2. On receiving the new directive, it confirms that either virtual actuator can pursue either directive on its own, and the tasks are split. If, on the other hand, P1 receives the same instruction to pursue, it reports that it cannot because patrolling requires both ground vehicles. C instructs P1 to pursue, which has higher priority, and registers P1's inability to patrol upon assuming the pursuit. Thus it tries to achieve the patrol task by calling on its other actuator, P2, with results analogous to the previous case. (In practice, the implementation shortcuts some of this interaction by tagging each initial instruction

with a priority.)

Finally, the individual agents receive their instructions through P1 or P2. When they attempt to perform them, the C-Kit Execution Interface dispatches them via what turn out to be actual local actuators. Should they encounter a reportable event such as success, failure, or their arrival at some specified state, they report it to P1 or P2. There the C-Kit Knowledge Interface deposits the report in P1 or P2's knowledge base as though it was its own activities that provoked it.

In many cases such interactions are the same ones exhibited by individualistic systems. When a new task arrives, an elected or otherwise designated agent leader might query constituents for action capabilities before reaching a decision, just as C1 compiles its capabilities from P1 and P2's. That the individual agents might communicate amongst themselves to make a decision in unison is not exclusive to individual architectures. C1 may very well consult its knowledge base concerning P1s and P2's individual utilities for a given course of action, its query passing through the C-Kit Knowledge Interface and finally the agents themselves via the communications interface. Is this any worse than individual agents spawning some centralized process to compile their votes and issue the outcome? If the two approaches are functionally equivalent then CA may be operationally superior, making such centralization explicit and running it within a well-understood single-agent process. The alternative should be treated as an interesting, younger, and open research area in the best cases and an inefficient ad-hoc solution in the worst.

A second observation is that none of the C-Agents, at any level, need be autonomous. Not only can the individual agents within follow advisable architectures, they could even consist entirely of human elements. That is to say, the C-Kit can be viewed as a sort of ``command console" where human operators receive reports from constituents, arbitrate them with goals, and issue queries or commands. The usefulness of such approaches has already been successfully illustrated without the use of the C-Kit, and in fact helped inform its design (Myers & Morley 2001).

7 Results

Sample Domain During development the C-Kit has been continuously tested within an autonomous robots domain. In typical scenarios, the agents in question must form and then execute group plans for patrolling arbitrarily defined areas and pursuing any targets they might find, based on their varying capabilities. Though this controller is being developed to eventually run on hardware, at this point experimentation occurs in the simulator depicted below.

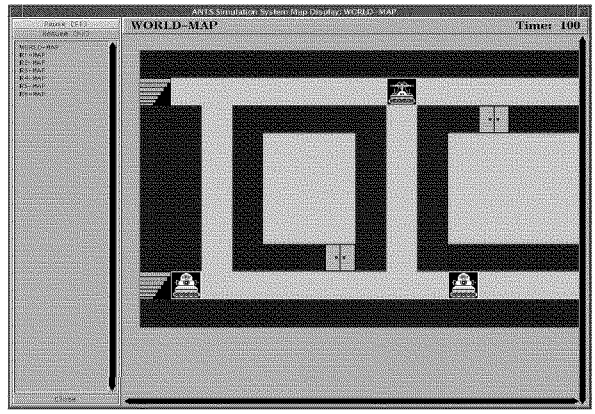

Figure 5: Software Domain Testbed.

Each robot has its own processor and memory, so in the experiments each agent runs on its own host, connected to all the others via a local-area network. Each C-Agent consists of an existing planning and control implementation coupled with a C-Kit encoded with domain knowledge in the form of plan templates, beliefs about each robot's capabilities, and rules for compiling information. During execution, the agents form a coalition in response to high-level directives to perform group tasks. Whenever this occurs, the system spawns a new C-Agent co-hosted with an arbitrarily selected team member.

The "PRS" Procedural Reasoning System, freely available from SRI International, provides the "existing" implementation for the C-Agents. It responds to mission orders or new perceptual information via the C-Kit's Knowledge Interface, and sends queries in the opposite direction. It then performs planning and scheduling, as well as execution monitoring, outputting requests for action to the C-Kit Execution Interface. Figure 6 depicts an example search plan represented in PRS's graphical control language. More detailed information on the PRS system can be found at http://www.ai.sri.com/~prs.

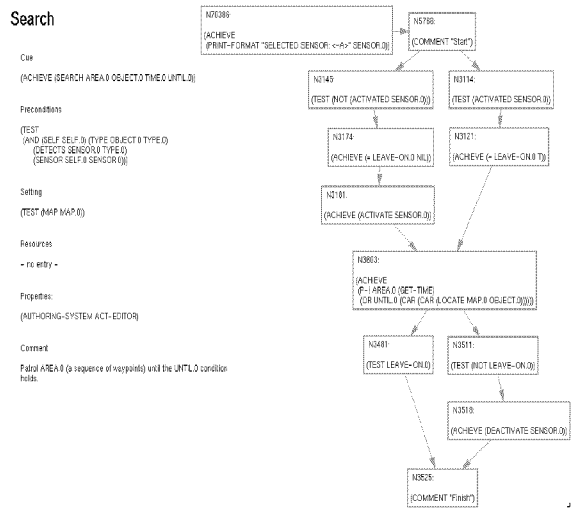

Figure 6: PRS Search Procedure

The completed system ran over 100 randomly generated scenarios in the sample domain, each requiring between two and five individual robots. The typical experiment called for three initial tasks for various combinations of agents to execute and plan concurrently. Then, during execution, two additional tasks would arrive, either as new mission directives from outside the system or via unpredictable events in the environment. To be more specific about the latter type, some initial tasks included instructions to initiate a new one should a certain condition come to pass. For instance, a group of agents might initially be instructed to patrol a particular area and then pursue any targets detected during the patrol.

Figure 7 presents some of the output generated during one such run. Here a higher-level C-Agent consists of two individual agents named "Cover" and "Point," and resides on the same host as Cover.

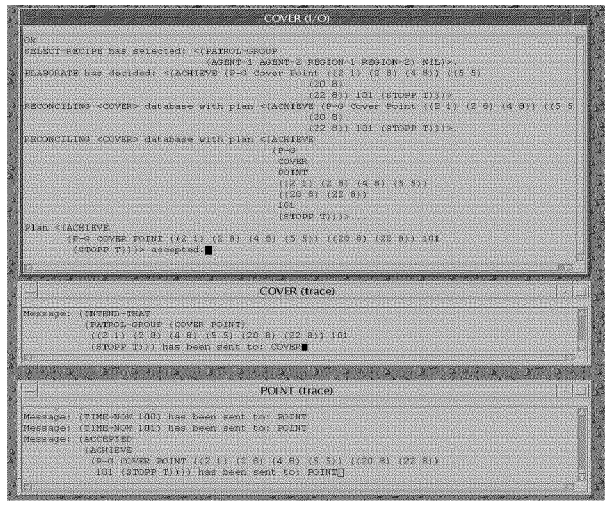

Figure 7: Example Execution Trace.

The top panel depicts the C-Agent superior generating of a group plan for patrolling a particular area, and the bottom two shows its individual instructions for action being sent to the two constituents.

Ignoring improved developmental efficiency, the test system should be evaluated for actual performance, which is here measured in terms of messages and execution speed. In the system, collective agents are allowed to deliberate for as long as it takes to receive responses to all their queries, so plan optimality is determined solely by the planning system operating separately in PRS.

For purposes of comparison, the scenarios were also run over nearly identical agents that were previously developed without the C-Kit. Such "individual" agents used the same straightforward knowledge compilation procedures, plan templates, and domain models as the collective ones, so they always arrived at and executed the same plans. Hence, the only difference was the organization of their communications and processing, thus isolating the characteristic consequences of the CA framework.

Table 2 compiles the results of such experiments. Each row specifies the performance of one of the systems over a certain class of scenarios, in terms of the total number of messages sent and the average amount of time elapsing between the arrival of a new task and its execution. In general, the CA system executed approximately 15% faster using almost 17% fewer messages when compared to the individualistic system over the course of all scenarios.

The most significant source of message conservation (and hence, execution speed) involved situations where the group arbitrators happened to reside on hosts that were centrally located on the network topology. Thus, they required few hops in to reach the constituents. In contrast, the individualistic agents sometimes had to send their messages across the entire network in order to coordinate their planning activities. This naturally suggests that high-level arbitrators should reside at network hubs.

	Messages	Time (ms)
C-Agents, All Scenarios	8,014	1.946
Individualistic, All Scenarios	9,580	2.285
C-Agents, Event-Driven only	4,586	2.140
Individualistic, Event-Driven only	4,736	2.221
C-Agents, Goal-Driven only	3,428	1.752
Individualistic, Goal-Driven only	4,844	2.344

Table 2: Comparison of Agent Types.

The data also illustrates that unfortunately, most gains arose within a specific class of scenarios, specifically those denoted as goal-driven scenarios. On the other hand, the two systems had nearly identical performance in scenarios that included event-driven goals.

Such directives require specific actions to take place once a particular condition holds, for instance "Stay still unless you detect a target" or "Patrol until fuel runs low." The problem for the C-Agents is that PRS performs such directives by checking the knowledge base each execution cycle in order to determine whether the condition has come true. With the individual agents, each agent performs such checks locally, and reports to the group only when new perceptual information affirms the targeted condition. On the other hand, when a C-Agent representing a group of robots pursues the same goal, the continuous stream of queries generated by its PRS cycle is dispatched as messages to constituents, through the Knowledge Interface.

Such goals only accounted for one fifth of the tasks in the category designated "Event-Drive" in the table. Otherwise, this problem would outweigh the other advantages enjoyed by the C-Agents and their performance would be far worse than comparable. This suggests that the formalism should be extended to allow notification hooks in constituents so that they can inform their superior arbitrators when a reportable event occurs. This would alleviate the need for constant querying and make the implemented system's behavior in such scenarios comparable to that of the individualistic agents.

Another complication deliberately excluded from the above examples is that sub-teams are not always disjoint from each other, and hence CA hierarchies need not always be trees. Referring once again to the example scenario, suppose that for some new task, each air vehicle must now work together with one of the ground vehicles. If this means that the patrols involving like vehicles have terminated, then P1 and P2 can be killed off and replaced by two new C-Agents representing the new working groups. If, however, both groups are to be maintained at once, then there will now be four upper-level C-Agents. This is not a problem for the system, as the formalism allows leaf C-Agents to work on multiple tasks, with contention to be resolved somewhere up the hierarchy where they have a mutual parent. That is, if the new patrols were under the control of a new coalition C2 that used some of C1's resources, a higher agent with C1 and C2 as children would resolve any contention.

The logistical problems caused by such complexity are not specific to CA, as individual agents would have to divide their attention and index their communications in accordance with the denser structure. However, the combinatorial blow-up arising as each possible grouping of agents comes into play is limited to increased processor time for individual agents. For C-Agents, though, there are memory considerations as well because a new C-Agent must begin operation for each new working group. Even if few domains lack enough natural structure to limit interaction hierarchically, the fact remains that C-Agents can run our of space in such applications.

Such results suggest that the Collective-Agents Framework is particularly well suited for large, widely distributed coalitions of operationally isolated teams. As individuals become increasingly numerous and far-flung, the conservation of communicative channels becomes increasingly important. At the same time, the number of extra arbitrators can never exceed the number of working groups; processing requirements are at most doubled in hierarchical cases. Further, agents working in distantly related groups produce more balanced tree structures, in turn limiting the number of extra hops between agents. Finally, developing such large networks exploits the uniform organizational structure of CA. Arbitrating processes can modularized and replicated across computational structures, and individuals need only report information and follow instructions.

On the other hand, smaller, more cohesive groups can perform better using individualistic methods. If every agent will at some point need to communicate with every other agent, and the group is compact enough that there

are plenty of channels available, then CA loses its communicative advantage over traditional approaches and begins to exhibit space problems. Further, as agents take on memberships in multiple groups, such problems are exacerbated. Hence, as an example, C-Agents might perform better as clearly delineated military entities than as specialized task forces. It should be observed that C-Agents can be used for the upper reaches of a hierarchy, with leaf C-Agents interfacing with alternative individualistic systems of agents rather than physical individuals.

8 Comparison

Most existing formalisms for multi-agent systems take an individualistic stance toward collaboration (Sonenberg et al. 1994; Cohen and Levesque 1990; Jennings 1995; Tambe 1997; Shoham 1993.) Somewhat surprisingly, we believe that the Collective-Agents framework is compatible with such approaches on two grounds. First, it is an implementational framework as well as a theoretical formalism. Secondly, it is nevertheless a philosophically principled move; it does not provide a software-based "hack" to approximate real-world phenomena.

In presenting their individualistic SharedPlans formalism, the authors observe that their logical constructions are not meant to be fed through a theorem-prover or serve as a software design document (Grosz and Kraus 1996.) Rather, formalisms specify group processes at a high level, serving to inform implementations that might take on very different forms. The CA framework presents a particular method for organizing agent implementations so that such high-level processes can be specified in an efficient, group-oriented manner. Whether individualistic agents adopting a new task elect a leader who polls them for conflicts with existing plans, or a high-level C-Agent arbitrator consults its constituents' databases via the Knowledge Interface, the patterns of messages traveling across the network are remarkably similar. At worst this is a new framework for specifying such patterns.

This is not to say that the motivations behind CA do not match its operational semantics. While individualism has long reigned in philosophical studies of intentional attitudes (Bratman, 1992; Searle 1990), a new movement has begun to explore collective mental phenomena as basic entities (Baier 1997; Gilbert 1996; Stoutland 1997.) Indeed, the creation of new arbitrators for group activity is much like Baier's mental commons. Such constructions need not exist physically to serve as useful conceptual schemes. For AI to be a valid enterprise, an entity's operation cannot be bound to our conceptual labels for its activities. Otherwise a computer cannot even add; it is merely moving electrical current through registers.

Ongoing work currently focuses on more efficient methods for event-driven behavior and on more intelligent protocols for hosting newly generated arbitrator processes. Another task is to try using the C-Kit with a more sophisticated planning system, such as SRI's SIPE family of planners built on PRS.

9 References

Amir, E. and McIlraith, S. (2000) "Partition-Based Logical Reasoning", Proceedings KR2000, pages 289-400.

Baier, A. (1997) "Doing Things with Others: The Mental Commons", In Alanen, Heinämaa, and Wallgren, editors, Commonality and Particularity in Ethics. St. Martin's Press, New York, pages 15-44.

Bratman, M. (1992) "Shared Cooperative Activity", Philosophical Review, 101(2):327-41.

Cohen, P. and Levesque, H. (1990) "On Acting Together", In Proceedings of AAAI-90:94-9.

Gilbert, M. (1996) "Walking Together: A Paradigmatic Social Phenomenon", Living Together: Rationality, Sociality, and Obligation. Rowman and Littlefield, New York.

Grosz, B. and Kraus, S. (1996) "Collaborative Plans for Complex Group Action", Artificial Intelligence, 86(1):269-357.

Jennings, N. (1995) "Controlling Cooperative Problem Solving in Industrial Multi-agent Systems Using Joint Intentions", Artificial Intelligence, 75(1):195-240.

McIlraith, S., Son, T.C. and Zeng, H. (2001) "Semantic Web Services", IEEE Intelligent Systems, Special Issue on the Semantic Web, 16(2);46-53.

Ortiz, C. L. and Hsu, E. I. (2002) "Structured Negotiation", To appear, Autonomous Agents and Multi-Agent Systems.

Ortiz, C. L., Hsu, E. I., DesJardins, M., et al. (2001) "Incremental Negotiation and Coalition Formation for Resource

Bounded Agents", Proceedings of the AAAI Symposium on Negotiation.

Searle, J. (1990) "Collective Intentions and Actions", In Cohen, Morgan, and Pollack, editors, Intentions in Communication. The MIT Press, Cambridge, MA, pages 15-32.

Sellars, W. (1968) "Science and Metaphysics: Variations on Kantian Themes", The Humanities Press, New York.

Shoham, Y. (1993) "Agent-Oriented Programming", Artificial Intelligence, 60(1):51-92.

Sonenberg, E. et al. (1994) "Planned Team Activity", In Castelfranchi and Werner, editors, Artificial Social Systems, Lecture Notes in Artificial Intelligence. Springer Verlag, Amsterdam.

Stoutland, F. (1997) "Why are Philosophers of Action so Anti-Social?", In Alanen, Heinämaa, and Wallgren, editors, Commonality and Particularity in Ethics. St. Martin's Press, New York, pages 45-74.

Tambe, M. (1997) "Agent Architectures for Flexible, Practical Teamwork", In Proceedings of AAAI-97: 22-8.